Sieťové aplikácie a správa sietí

Zadanie: Klient POP3 s podporou TLS

Peter Grofčík 1.11.2017

Obsah

1. Úvod

Úlohou bolo teda vytvorenie klienta komunikujúceho s POP3 serverom.
Prvá verzia projektu pracovala iba za použitia štruktúr BIO pre nadviazanie spojenia a komunikáciu, avšak táto varianta sa ukázala ako nekompatibilná pre vyriešenie IPv6 vzhľadom na to že BIO štruktúre nie je možné prideliť či pripojiť sa na základe IPv6 adresy, a preto v projekte kombinujem použitie raw socketu s BIO štruktúrou štýlom zabalenia socketu ako file deskriptora pod BIO štruktúru(a SSL štruktúru pre šifrované spojenia).

2. Vstupné argumenty

Pre vyriešenie vstupných argumentov som využil funkciu getopt(). Vytvorená class arg obsahujúca hlavnú funkciu arg zparsuje vstupné argumenty a vytvorí objekt pod danou classou s pre mňa dôležitými informáciami.

2.1 Nedefinovaná časť zadania

Pri nevyužití voliteľného argumentu -p [číslo portu] dôjde k implicitnému nastaveniu portu na hodnotu 110 alebo 995 na základe toho či bol použitý prepínač -T na šifrovanú komunikáciu alebo nie. Avšak prepínač -p má väčšiu prioritu ako implicitná voľba portu v programe a teda za predpokladu zvolenia zlého portu užívateľom je možné že sa nepodarí nadviazať šifrované či nešifrované spojenie na danom porte ak server na danom porte komunikuje ale nepodporuje danú variantu spojenia.

príklad:
pop.seznam.cz podporuje šifrované spojenie na porte 995
a nešifrované na porte 110 (tiež podporuje prechod na šifrované spojenie cez port 110 za použitia príkazu STLS)

Ak by teda došlo k pokusu o nadviazanie šifrovaného spojenia cez port 110, s najväčšou pravdepodobnosťou by došlo k problému pri connecte z dôvodu nepodareného handshaku.

Naopak ak by išlo o pokus o nadviazanie nešifrovaného spojenia cez port 995 zrejme by došlo ku connection timeout pretože by program neobdržal po connecte žiadnu odpoveď resp. by server posielal šifrované správy, ktorým by klient nerozumel.

Ďalším úplne nešpecifikovaným parametrom bol parameter -d. Za jeho použitia dôjde k odstráneniu všetkých práve stiahnutých správ.

Classa tiež obsahuje public funkcie pre získanie privátnych parametrov funkcie pre ďalšiu prácu programu.Za predpokladu že dôjde k akejkoľvek chybe (zle zadané argumenty) program vypíše nápovedu a ukončí sa .

3. Nadviazanie spojenia

Nadviazanie spojenia prebieha za použitia socketu a to na základe hostname alebo IP adresy servera zo vstupných argumentov.
Použitie socketu ešte pred komunikáciou cez BIO štruktúru pre mňa vyriešilo dva problémy v implementácii.

3.1 Timeout

V rámci BIO štruktúry nie je možné nastaviť časovač pre výpadok spojenia, a teda riešením pre mňa bolo nastavenie tohto timeoutu ešte pre socket a zabalenie socketu ako file descriptora pod BIO štruktúru pre ďalšiu prácu programu.

3.2 IPv6

Taktiež podľa informácii, ktoré som si dohľadal na internete nie je možné nadviazanie komunikácie cez BIO štruktúru na základe IPv6 adresy a teda k nadviazaniu komunikácie dochádza ešte pred zabalením socketu pod BIO štruktúru.

Pre nasledujúce spôsoby nadviazania spojenia som sa rozhodol na základe informácií z man stránok pri knižnice openssl a socketu, ale aj z rôznych informácií od užívateľov na stránke stackoverflow a informácii z referenčnej stránky ibm ohľadom možných/nemožných operácií s danými štruktúrami.
(napr. nemožnosť pripojenia na IPv6 za použitia BIO štruktúry – čo som vyriešil pripojením za použitia socketu a následným wrapnutím socketu pod BIO štruktúru).

Na implementáciu som nepoužil žiadne časti kódu z daných stránok. Jednotlivé kódy mi slúžili len k pochopeniu či rozhodnutiu použitia jednotlivých funkcii z openssl knižnice ak boli dané funkcie relevantné k mojej implementácií programu.

Po nadviazaní spojenia cez socket dôjde k nasledovným operáciám podľa zadaných vstupných argumentov:

3.3 Bez argumentov na šifrované spojenie

Dôjde k wrapnutiu socketu pod BIO štruktúru a nasledujúca komunikácia prebieha za použitia BIO_read() a BIO_write() funkcií.

3.4 Argument -T

Dôjde k nastaveniu SSL štruktúry pre šifrovanú komunikáciu. Následne dôjde k nahraniu parametrom zadaných certifikátov alebo defaultných ak neboli iné zadané.
Následne sa na komunikáciu používajú funkcie SSL_read() a SSL_write().

3.5 Argument -S

Klient odošle na server správu v tvare „STLS\r\n“ aby server vedel, že klient chce od tohto momentu komunikovať cez šifrované spojenie.
Za predpokladu odpovede -ERR zo servera je program ukončený pretože nie je možné prejsť z nešifrovaného spojenia na šifrované.
Následne dôjde k nastaveniu SSL štruktúry pre šifrovanú komunikáciu. Následne dôjde k nahraniu parametrom zadaných certifikátov alebo defaultných ak neboli iné zadané.
Od tohto momentu sa na komunikáciu používajú funkcie SSL_read() a SSL_write().

4. Autentifikácia

Za predpokladu, že nadviazanie spojenia prebehne bez chyby, program sa dostane do stavu autentifikácie, v ktorom sa program pokúsi autentifikovať voči serveru údajmi načítanými zo súboru určeného argumentom nachádzajúcim sa za argumentom -a.

Dochádza teda k odoslaniu správ na server v tvare:
USER [username zo súboru]
PASS [password zo súboru]

Po každej z týchto správ sa očakáva odpoveď zo server v tvare :
+OK
-ERR

V prípade oboch odpovedí v tvare +OK sa program nachádza v stave, v ktorom má prístup k obsahu schránky a teda môže pokračovať, ale v prípade že obdrží od servera správu -ERR dochádza k ukončeniu programu z dôvodu chyby autentifikácie na servery so zadanými prihlasovacími údajmi klienta.

5. Komunikácia bez voliteľných parametrov

Za predpokladu že boli použité len povinné parametre:
[hostname/IP adresa]
-o [adresár na uloženie správ]
- a [súbor s prihlasovacími údajmi]

V takomto prípade dôjde následne k stiahnutiu všetkých správ nachádzajúcich sa v klientskej schránke na servery.

Pre zistenie počtu správ nachádzajúcich sa na servery program odošle správu „STAT“. Následne očakáva odpoveď zo servera v tvare +OK [počet správ]. V prípade -ERR dôjde k ukončeniu programu.

Na základe tejto správy dochádza v loope k odosielaniu správy v tvare „RETR [i]“ pričom i je číselná hodnota generovaná v cykle od 1 po [počet správ].

Po každej odoslanej správe klient opäť očakáva odpoveď v tvare „+OK“ a po jej obdržaní dochádza k prímaniu obsahu správy (hlavička + text) až do okamihu kým prijatá správa neobsahuje na jej konci „\r\n.\r\n“ čo značí koniec primanej správy.

Následne môže program danú správu uložiť do nového či už existujúceho súboru s názvom:

[odosielateľ správy]_[dátum prijatia správy na server]

a pokračuje požiadavkou o ďalšiu správu.
Ak pri odpovedi na požiadavku obdrží v cykle -ERR, dôjde k ukončeniu programu.

6. Uloženie prijatej správy

Obsah uloženej správy je pre dodržanie formátu správ kompletným raw obsahom prijatej správy zo servera.

Názov súboru s uloženou správou môže byť v štyroch rôznych formách (prečo som sa rozhodol pre daný formát názvu je popísané v časti Voliteľný parameter -n):

[odosielateľ správy] – [dátum] (primárny formát):
Program sa pokúsi z prijatej správy vyzistiť odosielateľa a dátum. V prípade, že dôjde k zisteniu oboch informácií sú tieto informácie použité pre názov súboru.

„mail “ – [dátum]:
Formát názvu za predpokladu že dôjde k chybe pri dohľadaní odosielateľa správy.

[odosielateľ správy] – [UID správy]:
Formát názvu za predpokladu že dôjde k chybe pri dohľadaní dátumu.

„mail“ – [UID správy]:
Formát názvu za predpokladu že dôjde k chybe pri dohľadaní oboch informácií.

K chybe pri dohľadaní informácií potrebných na primárny názov súboru môže dôjsť za predpokladu že daná správa neobsahuje danú informáciu.
Správy neobsahujúce dané informácie nie sú klasické emaily, ale napr. serverom generované automatické správy (privítanie na servery pre nového užívateľa a pod.).

Týmito 4-mi formátmi som len ošetril variantu neklasických správ, aby nedochádzalo k chybe programu.K uloženiu správy nedochádza priamo do adresáru určeného argumentom -o.
Pre uloženie prijatej správy sa vytvára podpriečinok s dvomi úrovňami (za predpokladu, že ešte daný podpriečinok neexistuje):

/[out_dir]/[server]/[username]

Prijatá správa sa uloží do daného podpriečinku.

Daný spôsob podpriečinkov som zvolil z dôvodu prehľadnejšieho ukladania správ pre jednotlivých používateľov a pre jednotlivé servery. Taktiež pre ošetrenie prepisu uložených správ pre rôznych používateľov.

7. Voliteľné parametre

7.1 Voliteľné parametre -T a -S

Tieto vstupné argumenty upravujú len spôsob pripojenia na server. Zvyšok programu funguje rovnako či boli alebo neboli zadané.

Proces vykonania jednotlivých operácií za použitia niektorých z týchto parametrov som popísal v časti Nadviazanie spojenia.

7.2 Voliteľné parametre -c a -C

Tieto parametre môžu byť použité (nemusia) len v prípade, že bol použitý niektorý z parametrov -T alebo -S.
Parameter -c musí byť nasledovaný ďalším parametrom [súbor definujúci certifikát]. Pri jeho využití dôjde k pokusu o načítanie daného certifikátu, ale ak dôjde k chybe verifikácie certifikátu program je ukončený.

Parameter -C musí byť nasledovaný ďalším parametrom [priečinok s certifikátom]. Pri jeho využití dôjde k pokusu o načítanie certifikátu/certifikátov, ale ak dôjde k chybe verifikácie certifikátu program je ukončený.

Ak ani jeden z parametrov nie je použitý tak pred nadviazaním šifrovaného spojenia implicitne dôjde k načítaniu default certifikátov za použitia funkcie *SSL_CTX_set_default_verify_paths().*

A následne je na SSL štruktúre nastavený option za použitia flagu SSL_VERIFY_PEER pre implicitnú kontrolu certifikátu obdržanú zo servera pri handshaku po connecte.

7.3 Voliteľné parameter -p

Daný parameter musí byť bezprostredne nasledovaný číslom popisujúcim port na ktorý sa má klient pripojiť. Má väčšiu prioritu ako implicitné hodnoty 110 a 995 a teda môže dôjsť problémom popísaných v časti Vstupné argumenty -> Nedefinovaná časť zadania.

7.4 Voliteľný parameter -d

Pri jeho použití dôjde k odstráneniu práve stiahnutých správ zo servera.
Keďže počas aktívneho spojenia nedochádza k prepisu UID správ na servery je možné vždy po stiahnutí jeden správy odoslať požiadavku na odstránenie danej správy na servery bez toho aby došlo k chybe pri požiadavke na stiahnutie správy s UID+1.

7.5 Voliteľný parameter -n

Z naštudovaných informácií som dospel k záveru, že nie je možné implicitne zistiť na každom servery či je daná správa nová. Niektoré servery (napr. eva.fit.vutbr.cz) má v obsahu hlavičky správy riadok obsahujúci informáciu Status : [*].

Za predpokladu, že daná správa už bola pop3 klientom stiahnutá nachádza sa v správe Status: RO. Avšak nie každý server danú informáciu obsahuje resp. neexistuje presný formát označenia už prečítanej správy na servery.

Keďže pop3 klient bol primárne určený k stiahnutiu správ zo servera a ich následnému odstráneniu na servery a teda pri opätovnom spustení programu vždy došlo len k stiahnutiu nových správ, keďže staré sa na servery nenachádzali bola by to jedna možnosť riešenia, avšak zo zadania nemalo implicitne dochádzať k odstraňovaniu stiahnutých správ, ale iba ak bol program spustený s parametrom -d. Preto ani táto možnosť neprichádzala do úvahy pri mojej implementácií.

Do úvahy prichádzala možnosť uchovávania stiahnutého počtu správ a teda UID poslednej stiahnutej správy pri poslednom spustení. A teda pri opätovnom spustení a sťahovaní správ s parametrom -n by došlo len k stiahnutiu správ s vyšším UID. Táto možnosť bola veľmi ľahko realizovateľná keďže z informácií na fórum pre projekt bola informácia, že nemusíme riešiť prípadný prepis správ stiahnutých predošlým spustením, pri ktorom došlo aj k vymazaniu správ. Čo teda znamenalo že ak boli správy zmazané na servery tak pri následnom spustení s parametrom -n nemuselo dôjsť k stiahnutiu nových správ pretože po ukončení predošlého spojenia došlo k zmenám UID správ na servery (teoreticky teda správa na servery s najväčším UID mohla mať menšie UID ako to uložené).
Táto možnosť sa však z hľadiska použitia programu nepáčila mne, a preto som zvolil vlastné riešenie.

7.5.1 Riešenie na základe obsahu stiahnutých správ v adresári určenom parametrom -o.

Moje riešenie teda predpokladá, že ak skutočne chceme aby program stiahol len nové správy s parametrom -n, nesmie byť zmenený adresár pre uloženie správ. Ak k takej zmene dôjde budú stiahnuté všetky správy nachádzajúce sa na servery a neexistujúce v danom adresári (zrejme všetky).

Pre riešenie môjho výberu som pozmenil formát mena súboru do ktorého bude daná správa uložená na [odosielateľ] – [dátum].

Pri použití parametru -n dôjde k požiadavke o každú správu na servery a po jej prijatí následne overujem či daná správa na základe formátu mena už v danom priečinku existuje. Ak nie tak ju uložím a ak áno tak zahodím a pokračujem. Táto metóda je z hľadiska časovej náročnosti programu pomalšia ako metóda za použitia UID správ, avšak nedochádza k strate predtým stiahnutých a zo servera odstránených dát a to sa mi páči.
Pre zníženie časovej náročnosti programu som tiež implementoval príkaz TOP, vďaka ktorému sa mi podarilo časovú náročnosť dostať na úroveň využitia UID (za predpokladu že je TOP implementovaný na servery).I keď sa podľa RFC jedná o príkaz, ktorý na servery implicitne implementovaný byť nemusí, tak pri testovaní programu bol implementovaný na každom jednom, na ktorom som to testoval.
Pre prípad, že TOP na servery implementovaný nie je, som zaviedol testovací top. Za predpokladu že sa na servery nachádza aspoň jedna správa odošle sa implicitne príkaz v tvare „TOP 1 0“ čo má za následok odpoveď zo servera v tvare +OK alebo -ERR. Ak dôjde k odpovedi -ERR tak pri sťahovaní jednotlivých správ sa o použitie TOP príkazu nesnažím, aby nedošlo k odpojeniu servera z komunikácie z dôvodu chyby viacerých neznámych príkazov.

Ak však odpovie +OK tak pri sťahovaní jednotlivých správ daný príkaz používam, čo má za následok urýchlenie programu, keďže v prípade, že už dané správy existujú, nedochádza k stiahnutiu celej správy ale iba k stiahnutiu hlavičky o priemere 10-15 riadkov, ktoré trvá zanedbateľnú dobu oproti sťahovaniu celých niekedy naozaj veľkých správ.

8. Ukončenie spojenia

Nie vždy dôjde ku korektnému ukončeniu spojenia vzhľadom na to že môže dôjsť k výpadku serveru počas spojenia či vedľajšej strate dát.

Ku korektnému ukončeniu dochádza vždy za predpokladu že všetky operácie boli úspešne uskutočnené, či keď bola obdržaná -ERR odpoveď zo servera na niektorý z posielaných príkazov.

Formát korektného ukončenia je odoslaná požiadavka „QUIT“ a správna odpoveď +OK od servera po čom následne sa server odpojí, čo tiež spraví klient a program uvoľní všetky zabrané štruktúry BIO/SSL/socket.